TEMPEST version 6.0

Alfred Wong and Tom Pistor

Electronics Research Laboratory

University of California, Berkeley

1.0 Introduction

1.1 Overview

The acronym TEMPEST[3] stands for “Time-domain Electromagnetic Massively Parallel Evaluation of Scattering from Topography.” The computer program solves Maxwell’s equations using a time-domain finite-difference algorithm, where the electric and magnetic field nodes are spatially and temporally staggered over a three-dimensional topography of interest. Version 3.0 takes advantage of the inherent parallel nature of electromagnetic wave propagation and is implemented on the computer architecture connection machine 5 (CM-5)[8]. Due to the limited availability of the CM-5, version 4.0 is implemented on any single-processor computer architecture such as a work station or even a personal computer. Version 5.0 was also intended for single processor architectures but offered several improved features. Version 6.0 adds the Fourier Boundary Condition for the simulation of EUV multilayer mirrors and (ironically) reinstitutes the capability to run in parallel across several processors. The simulation domain may represent periodic, symmetric or isolated topographies. The algorithm is capable of simulating problems such as scattering from asymmetrical alignment marks, transmission through phase-shifting masks, effects of line-edge profiles in metrology and now the Fourier Boundary Condition and the reinstituted parallel capabilites, EUV multilayer mirrors and masks.

Illumination is assumed to be monochromatic, with the electric field linearly polarized in any user-specified direction. The incident angle can take on discrete values depending on the illumination wavelength and the dimension of the simulation domain. Illumination is assumed to be coherent and can consist of any intensity and phase profile such as that calculated from SPLAT9 or other computer programs.

TEMPEST parses topography information from an input file which can be checked for correctness. The input geometry is then simulated until the electromagnetic field reaches steady-state or, in the case of non-convergence, the simulation domain is excited for a user chosen number of wave cycles. Information on the simulation parameters is written to an output file. Topography and field data are written to files where they can be analysed.

1.2 New in Version 5.0

Version 5.0 of the program TEMPEST has several extensions from the previous version (version 4.0). First and foremost is TEMPEST's new ability to update nodes representing different materials with different updating equations. This allows more computation time and memory to be devoted to nodes which represent complicated materials (such as a dispersive metal) rather than wasted on nodes representing simple materials (such as air or glass).

This localization of the updating equations leads to a new implementation of boundary conditions. TEMPEST 5.0 treats boundary conditions merely as nodes with "special" updating equations. This conveniently allows the implementation of the newly invented boundary condition, the PML boundary condition[references].

Other side effect improvements:

PML boundary condition now allows the simulation of fully isolated topographies.

Excitation is no longer limited to a single planar distribution of excited nodes near the top of the domain. Plane sources with x,y and z directed normals and point sources can be placed anywhere in the domain.

Convergence checking considers the convergence status of a 3D grid of points distributed evenly throughout the domain rather than a single plane near the top. This helps to avoid false convergences.

Output data is no longer written in the PLOTMTV format as in version 4.0 but rather in a binary data format which requires much less disk space and is more portable.

Philisophical Changes:

Rather than writing output data (such as the steady state fields and refractive index) in the PLOTMTV format it is written in a binary format which uses much less space and can be read by a number of other programs. One such program is the MATLAB[ref] program. The new philosophy is to use MATLAB to analyse the data since MATLAB is a well developed, and widely used program. Several MATLAB script files have been developed for this purpose.

Move the TEMPEST-SPLAT interface out of TEMPEST and into MATLAB script files.

1.3 New in Version 6.0

1) Parallel Execution

The FDTD method, due its local updating equations, is easily parallelizable. TEMPEST 6.0 is capable of simulating large simulation domains using multiple processors. The parallel process communication library Message Passing Interface (MPI) is used to allow the multiple processors to communicate information between themselves.

When a parallel simulation is run, the rectangular simulation domain is divided up into smaller rectangular subdomains. Subdomains must communicate field and convergence information with neighboring subdomains. The user is responsible for specifying how the simulation domain is broken into the subdomains. This is done with the netlist file.

The netlist file is a text file which has a format that is best explained by example:

Netlist example:

A 600 x 300 x 200 (nx x ny x nz) simulation domain is to be subdivided into four subdomains of equal size (300 x 150 x 200).

The corresponding netilist will look like this:

```
4
0  0  299  0  149  0  199  1  1  2  2  0  0
1  300  599  0  149  0  199  0  0  3  3  1  1
2  0  299  150  299  0  199  3  3  0  0  2  2
3  300  599  150  299  0  199  2  2  1  1  3  3
```

The first line contains only one number, 4, which represents the number of subdomains.

The remaining lines describe the size and connectivity of each of the individual subdomains and have the following format:

i xmin xmax ymin ymax zmin zmax p_xmin p_xmax p_ymin p_ymax p_zmin p_zmax

where

i is the subdomain number,

xmin is the minimum cell number in the x direction,

xmax is the maximum cell number in the xdirection,

ymin is the minimum cell number in the y direction,

ymax is the maximum cell number in the y direction,

zmin is the minimum cell number in the z direction and

zmax is the maximum cell number in the z direction

p_xmin is the subdomain number of the subdomain connected to this subdomain's x=xmin boundary

p_xmax is the subdomain number of the subdomain connected to this subdomain's x=xmax boundary

p_ymin is the subdomain number of the subdomain connected to this subdomain's y=ymin boundary

p_ymax is the subdomain number of the subdomain connected to this subdomain's y=ymax boundary

p_zmin is the subdomain number of the subdomain connected to this subdomain's z=zmin boundary

p_zmax is the subdomain number of the subdomain connected to this subdomain's z=zmax boundary

Remember that periodic boundary conditions imply that the domain "wraps around onto itself" in each of the x, y and z directions. Thus, it is possible that the domain's x-max boundary neighbors its own x-min boundary.

Note that excitation planes and Fourier Boundary Condition planes can not be broken by subdomains. For this reason is is suggested that the user always divide the subdomain only with xy-planes. Here's what the netlist would look like using the previous example if we wanted to use only xy-dividing planes:

```
4
0  0  599  0  149  0  49  0  0  0  0  3  1
1  0  599  0  149  50  99  1  1  1  1  0  2
2  0  599  0  149  100  149  2  2  2  2  1  3
3  0  599  0  149  150  199  3  3  3  3  2  0
```

To run TEMPEST6.0 in parallel, use an additional command line argument as input to tempest:

`prompt% tempest` *file.in file.out file.net*

where *file.in* is the input file, *file.out* is the output file and *file.net* is the netlist file.

The exact command line to run the parallel simulation depends on many things including the type of machines, the particular MPI implementation, the file system structure and the operating system. For Solaris x86 running on pentiums with Argonne National Laboratories' MPI implementation the following command line works:

`prompt% mpirun -np 4 tempest` *file.in file.out file.net*

Each subdomain will, upon completion of the simulation, output its own .out file and any field information requested by the .in file. Filenames for files output by each subdomain will have a suffix which indicates which subdomain the file was written by. The usual MATLAB scripts (plotam.m plotbin.m etc.) can be used to view the output files.

2) Fourier Boundary Condition

The Fourier Boundary Condition allows the user to replace an entire 1-dimension structure (such as a film stack or multilayer mirror) with a boundary condition which provides equaivalent reflectivity. This works by decomposing the incident field into plane waves and applying an appropriate reflection coefficient to each of the incident plane waves. Due to the use of the discrete fourier transform (DFT), the FFTW library from M.I.T. (www.fftw.org) is required for compilation of the code. The reflection coefficient (versus angle of incidence is specified by the user either by a text file or by specifying material properties (n and k) for a material.

3) Miscellaneous changes

order_source - the **plane_source** command is replaced by the more useful **order_source** command. The **order_source** command allows the user to excite a unidirectional plane wave from an xy-plane called the “excitation plane”. Rather than specifying the angles of propagation (theta and phi), the “order” of the plane wave and the polarization is specified. This is a more natural specification of a plane wave since periodic boundary conditions imply that only a discrete set of angles of propagation are allowed. This eliminates the confusion associated with attempting to specify a non-propagating angle of incidence. (i.e. an angle which is not allowed by the periodic boundary conditions).

New format for cell coordinates: when using “node” units for specifying the dimensions/location of a primitive, previously (ver5.0) the coordinates were “inclusive” meaning that if xmin was 0 and xmax was 9 then the primitive would be 10nodes in the x-dimension. In ver6.0, this has been changed. To specify that the the primitive is 10 nodes in the x-direction, you must specify that xmin=0 and xmax=10. Note that this new rule is only applied to primitives and not output planes. The format for the output planes remains the same (i.e. xmin=0 xmax=99 means 100 nodes).

New notation “cells” instead of “nodes”: It was decided that the term “node” is misleading. “node” refers to a point in space - specifically a point where two or more objects are connected. This is misleading because it is being used to refer to a region of space (a cubic “cell”) inside which the fields are represented by a discrete set of numbers (Ex, Ey, Ez, Hx, Hy and Hz and perhaps flux components). Thus, the term “node” is being replaced by the term “cell”.

1.4 Applications Information

TEMPEST is capable of simulating a number of optical phenomena occurring in photolithography and optical metrology applications. These include reflective notching occuring in photoresist in the presence of non-planar topography, dark-field and bright-field imaging of wafer features (e.g., alignment marks and line structures), generation of image profiles through masks (e.g., reduction phase-shift mask), and calculation of the electric and magnetic fields at all points in the three-dimensional simulation domain at any instant in time. Several studies involving these topics have already been performed, and the user is encouraged to refer to these publications[2,3,6,7,11,12,13,14].

1.5 Source Code Information

TEMPEST ver6.0 is written in the C programming language. The source code is distributed into twelve text files. Six header files are also included. The header files are:

configuration.h	configuration constants
defn.h	constant definitions
global.h	global variable definitions
header.h	header files to be included
lib.h	libraries to include
routn.h	subroutines used in the program
ver.h	version and release date information

The files which contain the source code are:

bulk.c	interior updating equations
fbc.c	fourier boundary condition algorithm
in.c	reads information from the input file
init.c	initialization of variables
iterate.c	fields computation by iteration
loop.c	calls different loops according to the model
main.c	program control
math.c	mathematical functions
misc.c	miscellaneous functions
out.c	**prints simulation results to output files**
parproc.c	**communications for parallel processing**
pml.c	**prints simulation results to output files**
resist.c	**models photoresist bleaching**

Several MATLAB script files also have been written and are used to view the output data:

plotbin.m	view binary data
plotbinsten.m	view binary data and stencil in topography
plotam.m	view field time averaged amplitude
plotamsten.m	“ with topography stenciled in
plotam6.m	view total field
plotam6sten.m	“ with topography stenciled in

Please note that the source code is not complete without the MPI libraries from Argonne National Labs (www.anl.gov) and the FFT libraries from M.I.T. (found at www.fftw.org).

1.6 About this Manual

Section 1.0	**Introduction** A general overview of TEMPEST is provided.

Section 2.0	**Installation** This section discusses how the simulation program TEMPEST is compiled and installed.
Section 3.0	**Tutorial by Example** Two tutorial examples to let the user become familiar with the commands and procedures of running TEMPEST are given
Section 4.0	**Detailed Command Descriptions** Detailed descriptions of the commands, input file keywords and MATLAB script commands are presented in this section.
Section 5.0	**Useful Hints** Some useful hints for running TEMPEST are given.
Section A.0	**Suggested File Naming Convention**
Section B.0	**MATLAB script files**
References	

The conventions used throughout this users' guide are as follows:

- Words which are in ALL CAPITAL LETTERS represent programs or shell scripts.
- Words in **bold** letters represent commands to be typed exactly as it appears on the users' guide.
- Words in *italics* represent commands or arguments which can take on different values or character strings.
- Words inside [square brackets] are options to commands which may be omitted.
- Words in the `Courier font` represent characters in a file or output from programs.
- The string "ws%" represents the unix prompt on a workstation.
- The string "pc>" represents the DOS prompt on a PC.
- The string "prompt:" represents either and unis or DOS prompt.
- The string ">>" represents the MATLAB prompt.

2.0 Installation

2.1 Resource Requirements

2D simulations typically require anywhere from 1MByte to 20MBytes while 3D simulations typically require anywhere from 40MBytes to 1GByte. A 200MHz or faster processor is suggested.

TEMPEST is written in C and can be compiled on most machines. Due to the increased difficulty in compling the source code, executables for several machines are made available. Currently executables exist for PC's (i.e. Microsft Windows 98/NT/2000), SUN UltraSPARC II/Solaris and DEC-Alpha/HP-UX. See www.lava.eecs.berkeley.edu complete details.

MATLAB software can be used to view the output files (several MATLAB script files are also included in the package).

2.2 Compiling TEMPEST 6.0

When custom modifications are necessary, the source code must be modified and compiled. Compiling TEMPEST is complicated by the various compiler options available on the different machines, and also by the use of the following libraries:

1) For a parallel version of the code, an implementation of MPI (Message Passing Interface) must be present (see for example www-unix.mcs.anl.gov/mpi/index.html).

2) The discrete Fourier transform libraries FFTW (developed by Matteo Frigo and Steven G. Johnson from MIT - see www.fftw.org) must be also available.

A new header file, **configuration.h** contains C pre-processor directives which dictate which features will be compiled and for which platforms. The exact compilation procedure varies greatly from platform to platform and is not described in this User's Manual.

3.0 Tutorial by Example

The easiest way to learn how to use TEMPEST is by example. This chapter gives a brief overview of how to use TEMPEST and then presents an example simulation.

3.1 Overview

The basic steps in running TEMPEST simulation are shown as follows:

1) Edit input file with Text Editor

2) Run TEMPEST

3) Use MATLAB (or some other porgram) to view data files output by TEMPEST.

or

Post Process the output data (i.e. send output to an aerial imaging program such as SPLAT)

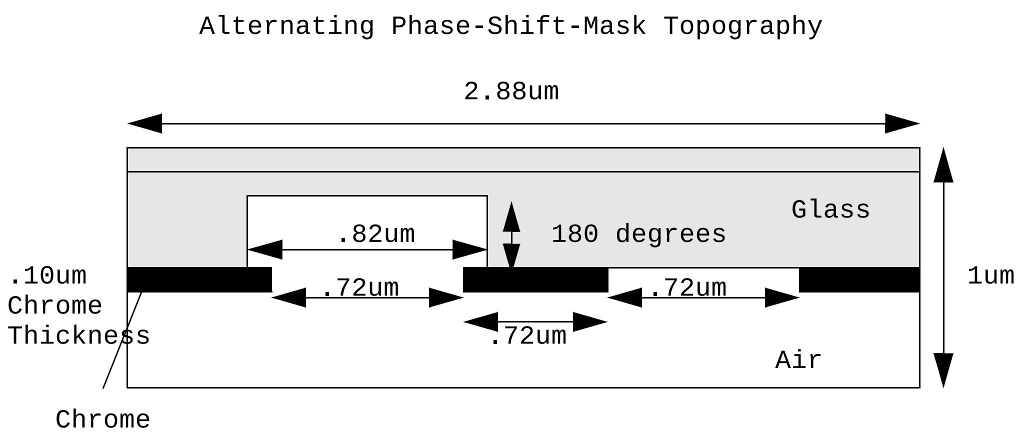
The input file to TEMPEST can be given any name and the entries in the input file can occur in random order. The input file to TEMPEST consists of a list of keywords; each keyword may be followed by numbers, words, or other keywords. A detail description of the keywords is found in Section 4.1. Comments can be included in the input file by including the string “/*” before the comment and the string “*/” after the comment.

To execute the program TEMPEST, the user must supply two arguments: the first being the name of the input file, and the second is the name of the output file to which information concerning the simulation is to be written.

The amount of memory needed to run the program is proportional to the number of simulation nodes in the domain, i.e., the product of the number of nodes in the x-, y-, and z-directions. As a rule of thumb, a domain with 1 million nodes (100 by 100 by 100) requires about 30 MBytes of memory (30 bytes/node). A simulation can take anywhere from a few minutes to several hours depending on the size of the problem.

3.2 Example - Alternating Phase Shift Mask Line/Space Pattern

The first example is that of an alternating phase shift mask line/space pattern. The wavelength is 248nm, the CD (line and space widths) is 180nm. The simulation domain and topography are depicted in Figure 1.

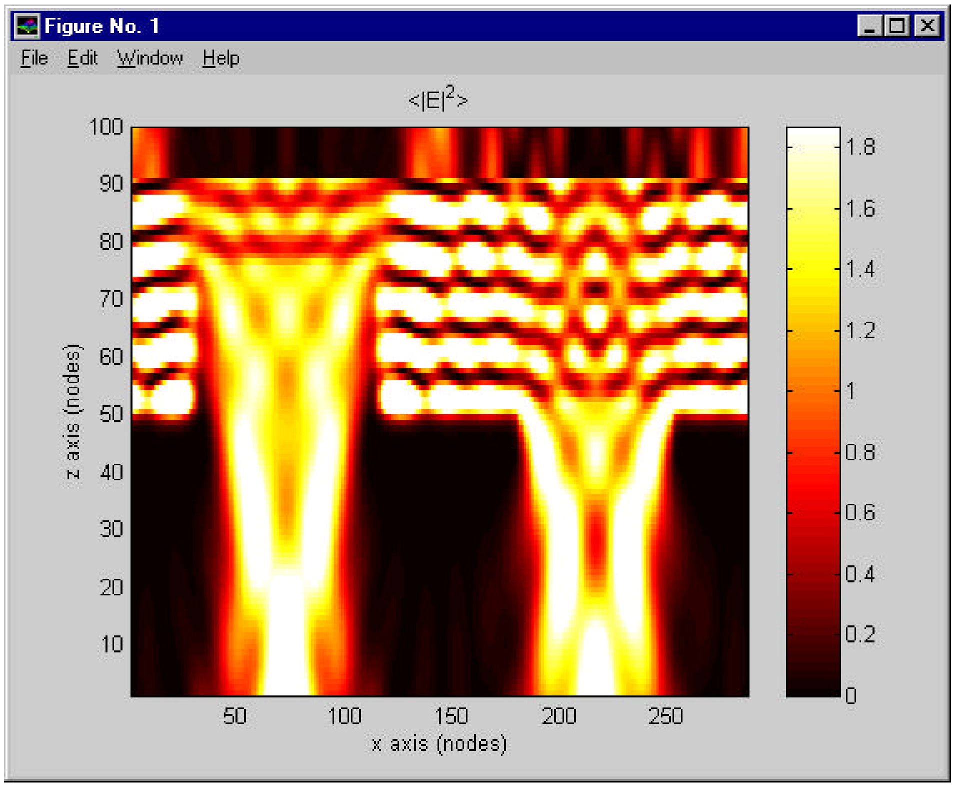


```
/*
LineSpace.in
TEMPEST input file for Alternating Phase Shift Mask Topography
*/
wavelength 0.248     /* wavlength is 248nm */
x_node 288           /* # cells in domain x-dimension */
y_node 1             /* # cells in domain y-dimension - this is a 2D sim */
z_node 116           /* # cells in domain z-dimension */
x_dim 2.88           /* Length of the domain x-dimension */
                     /* dx=dy=dz will be calculated by TEMPEST as x_dim/x_node */
                     /* y_dim, and z_dim are also calculated by TEMPEST - no need
                       to specify them. */

/* Excite a normally incident plane wave travelling in the downward (-z) direction */
order_source_nk node 98 -1 te 0 0 1.0 0.0 1.50841 0.0

/* First block fills the domain with air */
rectangle position 0.0 2.88 0.0 0.01 0.08 1.08 index 1.0 0.0

/* Next block fills the top half of the domain with glass */
rectangle position 0.0 2.88 0.0 1.0 0.58 1.08 index 1.50841 0.0

/* Next the chrome is placed underneath the glass.  Chrome has k>n and so
   the 'dispersive' command must be used */
rectangle position 0.0 2.88 0.0 1.0 0.48 0.58 dispersive 0.85 2.01

/* The next three blocks are air and form the unshifted line and the
   phase-shifted line */
rectangle position 0.36 1.08 0.0 1.0 0.48 0.58 index 1.0 0.0
rectangle position 1.8 2.52 0.0 1.0 0.48 0.58 index 1.0 0.0
rectangle position 0.31 1.13 0.0 1.0 0.58 0.8239 index 1.0 0.0

/* 8 cells of PML (matched to air) at the bottom of the domain form the absorbing
   boundary condition at the bottom of the domain */
rectangle node 0 288 0 1 0 8 pml 0 0 -1 1.0 1 0 0

/* 8 cells of PML (matched to glass) at the top of the domain form the absorbing
   boundary condition at the top of the domain */
rectangle node 0 288 0 1 108 116 pml 0 0 1 2.2753007281 1 0 0

/* Output the Ey field everywhere in the domain (the y=0 plane) */
```

```
plot ey steady 0.00 node 0 287 0 0 8 107 LineSpace.y.0.e.y.i
plot ey steady 0.25 node 0 287 0 0 8 107 LineSpace.y.0.e.y.q

/* Also plot the refractive index */
plot refractive node 0 287 0 0 8 107 LineSpace.y.0.n
```

To view the output from the TEMPEST simulation, MATLAB and the plotam.m script can be used.

```
>> plotam(‘LineSpace.y.0.e.y.i’,’LineSpace.y.0.e.y.q’);
```



4.0 Detailed Command Descriptions

4.1 Input File Keywords

The input file to TEMPEST contains five types of specifications:

1. simulation domain
2. illumination
3. topography
4. analysis (numerical parameters)
5. data (output specifications)

The input file to TEMPEST can be given any name (see Appendix A for suggested file naming conventions) and the entries in the input file can occur in random order. Comments can be included in the input file by including the string “/*” before the comment and the string “*/” after the comment. Further, this version (as does the previous version) of TEMPEST performs error checking of the input file and will be able to detect most input errors. The input file to TEMPEST consists of a list of keywords. Each keyword may be followed by numbers, words, or other keywords. The following is a description of the keywords:

4.1.1 Domain

x_node *x_simulation_nodes*

x_simulation_nodes is a number specifying the number of simulation nodes in the x-dimension of the simulation domain. *x_simulation_nodes* must be a positive number greater than or equal to 2. See below for restrictions.

y_node *y_simulation_nodes*

y_simulation_nodes is a number specifying the number of simulation nodes in the y-direction of the simulation domain. *y_simulation_node* must be a positive number greater than or equal to 1. See below for restrictions.

z_node *z_simulation_nodes*

z_simulation_nodes is a number specifying the number of simulation nodes in the z-direction of the simulation domain. *z_simulation_nodes* must be a positive number greater than or equal to 2. See below for restrictions.

x_dim *x_length*

x_length is the x-dimension of the simulation domain in micrometers. See below for restrictions.

y_dim *y_length*

y_length is the y-dimension of the simulation domain in micrometers. See below for restrictions.

z_dim *z_length*

z_length is the z-dimension of the simulation domain in micrometers. See below for restrictions.

Restrictions exist on the values *x_length*, *x_simulation_nodes*, *y_length*, *y_simulation_nodes*, *z_length* and *z_simulation_nodes*: TEMPEST 5.0 uses a cubic discrization grid placing the restriction that $\Delta x=\Delta y=\Delta z$ which means that *x_length*/*x_simulation_nodes* = *y_length*/*y_simulation_nodes*=*z_length/ z_simulation_nodes*. Consequently it is not necessary to specify all six parameters but rather only enough to completely define all three dimensions of the domain and the discretization Δx. Any unspecified parameters will be deduced by TEMPEST. If more than the minimum number of parameters are specified then they must satisfy the above restriction (to within numerical error) otherwise TEMPEST will give an error. Also, for 2D simulations the dimension in which there exists only one node must be chosen as the y-dimension.

x_symmetry

keyword **x_symmetry** specifies that the simulated topography has mirror symmetry in the x direction. Nodes with coordinates (0,y,z) will use nodes with coordinates (1,y,z) as their west neighbors and nodes with coordinates (*x_simulation_nodes-1*,y,z) will use nodes with corrdinates (*x_simulation_nodes*-2,y,z) as their east neighbors. If **x_symmetry** is not specified then periodic boundary conditions apply in the x-dimension where nodes with coordinates (0,y,z) use nodes with with coordinates (*x_simulation_nodes*-1,y,z) as their west neighbors and nodes with coordinates (*x_simulation_nodes*-1,y,z) use nodes with coordinates (0,y,z) as thier east neighbors.

y_symmetry

keyword **y_symmetry** specifies that the simulated topography has mirror symmetry in the y direction. Nodes with coordinates (x,0,z) will use nodes with coordinates (x,1,z) as their south neighbors and nodes with coordinates (x,*y_simulation_nodes-1*,z) will use nodes with corrdinates (x,*y_simulation_nodes*-2,z) as their north neighbors. If **y_symmetry** is not specified then periodic boundary conditions apply in the y-dimension where nodes with coordinates (x,0,z) use nodes with with coordinates (x,*y_simulation_nodes*-1,z) as their south neighbors and nodes with coordinates (x,*y_simulation_nodes*-1,z) use nodes with coordinates (x,0,z) as thier north neighbors. If *y_simulation_nodes* is 1, for example in a 2D simulation, y_symmetry will have no effect and all nodes will use themselves as their both their north and south neighbors.

z_symmetry

keyword **z_symmetry** specifies that the simulated topography has mirror symmetry in the z direction. Nodes with coordinates (x,y,0) will use nodes with coordinates (x,y,1) as their down neighbors and nodes with coordinates (x,y,*z_simulation_nodes*-1) will use nodes with corrdinates (x,y,*z_simulation_nodes*-2) as their up neighbors. If **z_symmetry** is not specified then periodic boundary conditions apply in the z-dimension where nodes with coordinates (x,y,0) use nodes with with coordinates (x,y,*z_simulation_nodes*-1) as their down neighbors and nodes with coordinates (x,y,*z_simulation_nodes*-1) use nodes with coordinates (x,y,0) as thier up neighbors.

4.1.2 Illumination

wavelength *lambda*

lambda is the free space wavelength of the incident illumination in micrometers. TEMPEST assumes monochromatic illumination.

point_source *unit x_pos y_pos z_pos xcomp ycomp zcomp mag pha*

unit can be either **position** or **node**. If **position** is specified then *x_pos, y_pos* and *z_pos* are the x, y and z coordinates in micrometers of a point source excitation. If **node** is specified then *x_pos, y_pos* and *z_pos* are the x, y and z nodal coordinates of the point source. The amplitude, phase and polarization of the point source are determined by the remaining parameters with the following formula:

$$\vec{E}_{ptsrc} = Re(mag e^{jpha}(xcomp \cdot \hat{x} + ycomp \cdot \hat{y} + zcomp \cdot \hat{z})) \quad \text{(EQ 1)}$$

plane_source *orientation pos_unit min1 max1 min2 max2 pos3 xcomp ycomp zcomp intensity pha type arg1 arg2*

This keyword specifies the existence of a planar segment excitation. *orientation* can be one of {**xy**, **yz** or **zx**} corresponding to xy, yz or zx planar segments of excited nodes. *pos_unit* can be either **position** or **node** corresponding to the units (micrometers or node number) of the next five entries which determine the size and position of the planar segment source. *min1* and *max1* specify the extent of the source in the 1st direction and *min2* and *max2* specify the extent of the source in the 2nd direction while *pos3* specifies the position of the source in the 3rd direction. See table below for an explicit description.

TABLE 1. Table of Coordinates for the various Planar Segment Source Orientations

Orientation	Normal	min1,max1 direction	min2,max2 direction	pos3 direction
xy	z	x	y	z
yz	x	y	z	x
zx	y	z	x	y

intensity is the intensity in mW/cm^2 of the source if it were a normally incident plane wave in free space. The electric field is calculated from the following formulas:

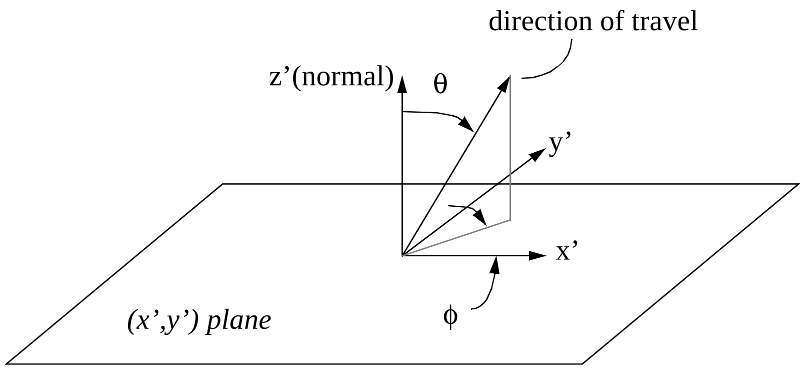
$$E_x(x', y') = xcomp \times a(x', y')e^{j(\phi(x', y') + pha)} \times \sqrt{2 \times value \times 10 \times \eta_{freespace}}$$

$$E_y(x', y') = ycomp \times a(x', y')e^{j(\phi(x', y') + pha)} \times \sqrt{2 \times value \times 10 \times \eta_{freespace}}$$

$$E_z(x', y') = zcomp \times a(x', y')e^{j(\phi(x', y') + pha)} \times \sqrt{2 \times value \times 10 \times \eta_{freespace}} \quad \text{(EQ 2)}$$

where $a(x', y')$ and $\phi(x', y')$ are magnitude and phase functions determined by the remaining arguments. *type* can be **uniform** or **file** meaning that the distribution of intensity accross the (x', y') plane is uniform or nonuniform and read in from a file on disk.
If **uniform** is selected then the distribution is uniform and represents a plane wave travelling in the direction specified by *arg1* and *arg2*. *arg1* is the θ measured from the positive normal to the (x', y') plane. *arg2* is the azimuthal angle ϕ measured from the x' axis in the counterclockwise direction. See diagram below:

Uniform Plane Wave Direction of Travel



If **file** is selected then the distribution is loaded in from disk. *arg1* and *arg2* are the filenames of the magnitude $a(x', y')$ and phase $\phi(x', y')$ files respectively.

order_source *unit z_pos dir polarization m n mag pha*

unit can be either **position** or **node**. If **position** is specified then *z_pos* is the z coordinate micrometers of an xy-plane excitation. If **node** is specified then *z_pos* the z nodal coordinate of the excitation plane. *dir* indicates the direction of propagation (+1 means the excitated field propagates up and -1 means the excited field propagates down). *polarization* can be either **te** or **tm** and *mag* and *pha* are the magnitude (in V/um) and phase (at (x,y)=(0,0), in radians) of the excited field. *m* and *n* specify the “order” number and the following two formulae apply:

$$k_x = \frac{2\pi m}{L_x} \text{ and } k_y = \frac{2\pi n}{L_y}.$$

order_source_nk *unit z_pos dir polarization xorder yorder mag pha n k*

Same as **order_source** except that n and k specify the index of the material in which the exciation plane is contained.

4.1.3 Topography

The topography of the simulation domain (including the pml, first order and perfectly conduction boundary conditions) is specified with a series of block definitions. Each block definition has the following form:

primitive unit [pos1 pos2 pos3 . . .] material [arg1 arg2 arg3 arg4 . . .]

The number of positioning arguments (*pos1*, *pos2* etc.) depends on the primitive chosen and the number of material arguments (*arg1*, *arg2* etc) depends on the material chosen. In all cases *unit* can be either **position** or **node** specifing whether the positioning arguments are in micrometers or nodes (nodes are always numbered starting from zero) and *primitive* can be selected from any of the following primitives:

rectangle *unit xl xh yl yh zl zh material arg1 arg2 arg3 arg4 ...*

specifies that the existence of rectangular box bounded in the x-direction by [*xl*, *xh*], in the y-direction by [*yl*, *yh*], and in the z-direction by [*zl*, *zh*].

sphere *unit radius xcenter ycenter zcenter material arg1 arg2 arg3 arg4 ...*

specifies the existence of a sphere with radius *radius* centered at (*xcenter*, *ycenter*, *zcenter*).

xcylinder *unit radius xl xh ycenter zcenter material arg1 arg2 arg3 arg4 ...*

specifies the existence of a circular cylinder oriented along the x-direction from *xl* to *xh* with radius *radius*. The circular cross-section on the yz-plane is centered at (*ycenter*, *zcenter*).

ycylinder *unit radius yl yh zcenter xcenter material arg1 arg2 arg3 arg4 ...*

specifies the existence a circular cylinder oriented along the y-direction from *yl* to *yh* with radius *radius*. The circular cross-section on the zx-plane is centered at (*zcenter*, *xcenter*).

zcylinder *unit radius zl zh xcenter ycenter material arg1 arg2 arg3 arg4 ...*

zcylinder specifies the existence of a circular cylinder oriented along the z-direction from *zl* to *zh* with radius *radius*. The circular cross-section on the xy-plane is centered at (*xcenter*, *ycenter*).

xwedge *unit xl xh y1 z1 y2 z2 y3 z3 material arg1 arg2 arg3 arg4* ...

xwedge specifies the existence of a wedge with a triangular interface with the yz-plane defined by the coordinates (*y1*, *z1*), (*y2*, *z2*), and (*y3*, *z3*). The wedge is oriented along the x-direction from *xl* to *xh*.

ywedge *unit yl yh z1 x1 z2 x2 z3 x3 material arg1 arg2 arg3 arg4* ...

ywedge specifies the existence of a wedge with a triangular interface with the zx-plane defined by the coordinates (*z1*, *x1*), (*z2*, *x2*), and (*z3*, *x3*). The wedge is oriented along the y-direction from *yl* to *yh*.

zwedge *unit zl zh x1 y1 x2 y2 x3 y3 material arg1 arg2 arg3 arg4* ...

zwedge specifies the existence of a wedge with a triangular interface with the xy-plane defined by the coordinates (*x1*, *y1*), (*x2*, *y2*), and (*x3*, *y3*). The wedge is oriented along the y-direction from *zl* to *zh*.

material can be selected from the following material types:

index *n k*

The block represents an isotropic, linear, non-dispersive, non-magnetic material (such air, glass, resist, silicon are at optical wavelengths). n is the refractive index of the material and k represents the lossiness of the material (k=0 implies lossless material). give formula involving n-jk)^2=epsilon complex = . For this material, mu = m0=1.whatever

dispersive *n k*

same as index except used when k>n. yadda yadaa yada

cond *conductivity*

perfect conductor approximation ....

epsmu *eps_real eps_imag mu_real mu_imag*

gindex exx exy exz eyx eyy eyz ezx ezy ezz

sexx sexy sexz seyx seyy seyz sezx sezy sezz

mxx mxy mxz myx myy myz mzx mzy mzz

smxx smxy smxz smyx smyy smyz smzx smzy smzz

-not working yet

bc *orientation n theta*

black_matter

The fields inside a block with this material specification are zero. This can be used as a perfect electric/magnetic conductor boundary condition. The can also be used in regions of the domain which in which the fields are know to be zero (such as deep inside a metal object) to save computation time.

pml *xabs yabs zabs eps mu sigmae sigmam*

The PML (perfectly matched layers) is a non-physical anisotropic material which has the special characteristic that it can absorb incident radiation from various angles without reflection. It is used in TEMPEST as an absorbing boundary condition. *xabs, yabs* and *zabs* indicate the direction of travel the incident field to be absorbed has. They can take on the values -1, 0 or +1. See figure below.

Placing PML around the edges of the Simulation Domain

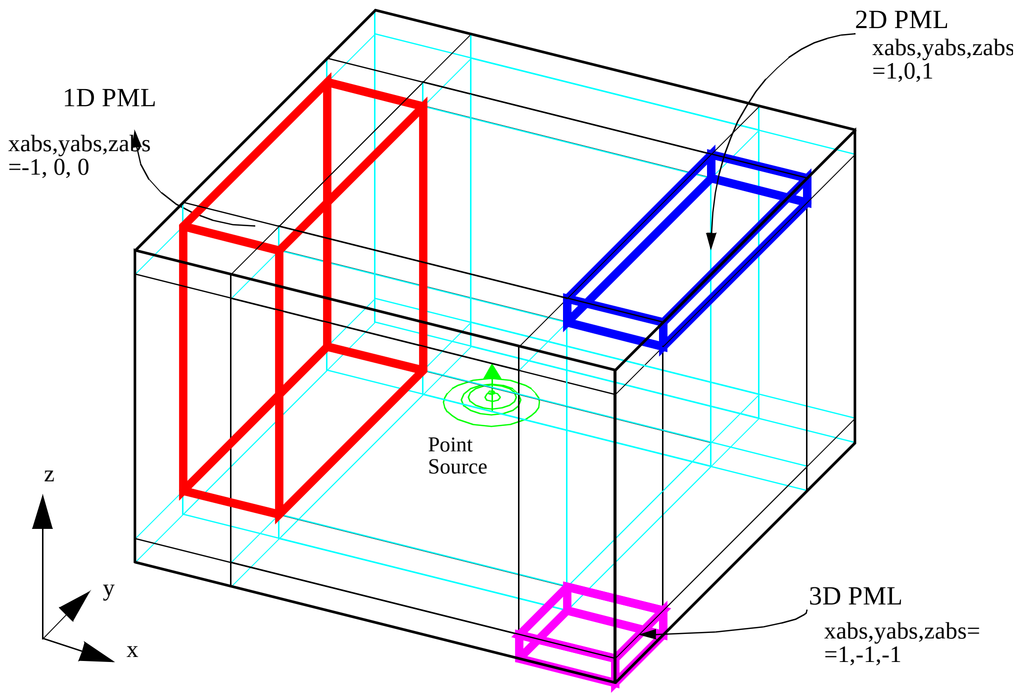
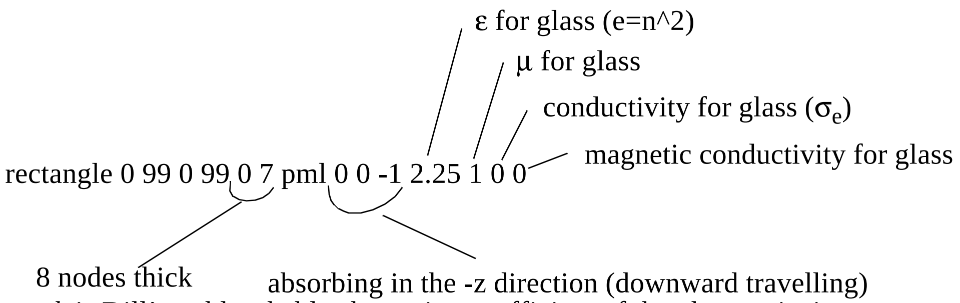


Figure 6. The PML material is placed around the domain. Ideally, all outgoing radiation (from the point source at the center of the box) is absorbed by the PML without reflection. Effectively the solution for the radiating point source in free space is found for the finite region of space modelled by the domain. Note that PML material is different (in its direction(s) of attenuation) for each face, edge and corner of the domain.

eps, *mu*, *sigmae* and *sigmam* are the ε, μ, σ_e and σ_m for the material adjacent to the PML material (from which the incident field to be absorbed by the pml comes). For example: A layer of PML at the bottom of the simulation domain adjacent to glass (n=1.5, k=0) would have an entry as follows:

resist *init_conc a b c*

resist specifies that the current block has dynamic change with respect to exposure. *init_conc* is the initial PAC value of the photoresist block. *a* is Dill’s bleachable absorption coefficient of the photoresist in per micro-meter.


ε for glass (e=n^2)
μ for glass
conductivity for glass (σ_e)
magnetic conductivity for glass
rectangle 0 99 0 99 0 7 pml 0 0 -1 2.25 1 0 0
8 nodes thick
absorbing in the -z direction (downward travelling)

b is Dill’s unbleachable absorption coefficient of the photoresist in per micro-meter. *c* is the bleach rate of the photoresist model proposed by Dill in centi-meter squared per milli-Joule.

fbc *unit z_pos index_or_file arg1 arg2*

fbc specifies the existance of a Fourier Boundary Condition situated at z=*z_pos* in the units specified by *unit*. *unit* can be either **position** or **node,** and *index_or_file* can be either **index** or **file**. If **index** is used, the FBC will be programmed to simulate a half-space (z<z_pos) of material with index (n and k) specified by *arg1* and *arg2*. If **file** is used, then the reflection coefficients (both TE and TM) are loaded in from a file which is specified in *arg1* (*arg2* is not used).

The reflection coefficient file is actually two files - one for the reflection coefficient magnitude and one for the reflection coefficient phase. The first file has a .mag file name extenstion (i.e. arg1.mag is the file name) and is a binary file (i.e. not text) with the following format:

```
(int) 91
(int) 1
(int) 1
(float) mag(rte(0 degrees))
(float) mag(rtm(0 degrees))
(float) mag(rte(1 degrees))
(float) mag(rtm(1 degrees))
(float) mag(rte(2 degrees))
(float) mag(rtm(2 degrees))
.
.
.
(float)mag(rte(90 degrees))
(float)mag(rtm(90 degrees))
```

The second file contains the phase information (file name is arg1.pha) and has a similar binary format:

```
(int) 91
(int) 1
```

```
(int) 1
(float) arg(rte(0 degrees))
(float) arg(rtm(0 degrees))
(float) arg(rte(1 degrees))
(float) arg(rtm(1 degrees))
(float) arg(rte(2 degrees))
(float) arg(rtm(2 degrees))
.
.
.
(float)arg(rte(90 degrees))
(float)arg(rtm(90 degrees))
```

Example 1: A FBC representing a half space of glass (n=1.5, k=0.0) from z=1 and below will have the following command format:

```
fbc position 1.0 index 1.5 0.0
```

Example 2: A FBC representing a multilayer mirror (who's reflection coefficients vs. angle is loaded from the file mm40.rc) which begins at z=2um (i.e. the mirror surface is at z=2um, and the mirror resides below z=2um) has the following command format:

```
fbc position 2.0 file mm40.rc
```

4.1.4 Analysis

min_cycle *min_wave_cycles*

This sets the minimum number of cycles TEMPEST will run to *min_wave_cycles* regardless of the convergence status. If unspecified, the default value is zero.

max_cycle *max_wave_cycles*

Simulation continues until steady-state is reached or until the domain has been excited with *max_wave_cycles* wave cycles, whichever comes first. If unspecified, the program will run until convergence occurs.

dt *value*

specify the time step to be used in seconds. If unspecified, TEMPEST automatically calculates the time step to be used based on .....

err_tol *fraction*

Steady-state is determined by comparing the field values at evenly distributed set of test points at the same instant in successive wave cycles. If the relative variation of the electric field intensity at each node in the test grid is less than fraction for three consecutive wave cycles, steady-state is reached. A typical value for fraction is 0.1 and the relative variation is defined:

$$relerror = \frac{|\vec{E}_{new}| - |\vec{E}_{old}|}{|\vec{E}_{new}| + |\vec{E}_{old}|} \qquad \text{(EQ 3)}$$

unless the denominator is deemed extremeley small in which case the point is not included in the comparison.

4.1.5 Data

plot *plot_var [arg1 arg2 . . .] coord_unit xmin xmax ymin ymax zmin zmax file*

plot_var must be one of the keywords: **refractive, block, ex, ey, ez, hx, hy, hz** or **pac**.

refractive requests a plot of the refractive index of the materials in the domain. A value of zero is reported for PML or 1st order boundary condtion materials. For anisotropic materials, the maximum refractive index is reported.

block requests a plot of the block number. Blocks are numbered starting from zero in the order of appearance in the input file.

ex, ey, ez, hx, hy, or **hz** request a plot of the corresponding electromagnetic field component. Two additional parameters, *arg1* and *arg2* must be specified. *arg1* must be one of **steady**, **nonsteady** or **transient**.

steady requests a field plot at the end of the simulation when either the fields have reached steady state or the maximum number of iterations (set by the **max_cycle** command) has been reached. The second argument, *arg2*, must be a number between 0 and 1 which specifies at what fraction of the wave cycle (the phase) the instantaneous field should be reported.

nonsteady requests a field plot at the instant of time specified by *arg2* (in #'s of wave cycles since the start of the simulation).

transient - not done yet.

pac - not done yet.

coord_unit must be either **position** or **node** indicating whether the following six coordinates are specified in micrometers or node numbers.
xmin, xmax, ymin, ymax, zmin and *zmax* specify what part of the domain will be plotted.
file is the filename on disk to write the data. If the file exists, it will be overwritten. See Appendix A for suggested file naming conventions.

orders *coord_unit zpos dir filename*

This command is used to output the orders (plane wave decomposition of the field) to a file. *coord_unit* must be either **position** or **node** indicating the units of *zpos*, the z position of the observation plane at which the field is decomposed into TE and TM plane waves. *dir* indicates which way the energy is assumed to be travelling and is either +1 (up, or +z direction) or -1 (down, or -z direction). The orders are output to the file given by *filename*. The order output file can be viewed with the vieworders MATLAB script.

4.2 TEMPEST command

The program TEMPEST takes two arguments. The first of which is the name of the input file which contains the topography information. The second argument is the name of the output file to which information on the simulation run is written.

prompt: **tempest** *input_file output_file*

See Appendix A for suggested file naming convention.

4.3 Post-TEMPEST Commands

Version 5.0 no longer supports the PLOTMTV format. All output data (with the exception of the standard output file automatically generated) is written in a binary data format which uses much less space and is easily ported. Several MATLAB scripts have been developed to view and/or modify the output data. The script files are given in Appendix B. Detailed command descriptions (from the MATLAB prompt >>) follow:

>>**plotbin(‘***filename***’);**

This command calls the plotbin.m script file. It is used to plot refractive index plots, block number plots and field plots at a particular instant of time.

>>**plotbinsten(‘***filename***’,’***top_filename***’)**;

This command calls the plotbinsten.m script file. It is similar to the **plotbin** command above but it stencils in the topography borders. *filename* is a field file (such as the x component of the electric field at steady state) and *top_filename* is a plot file which either contains the refractive index or block number of region corresponding that of *filename*.

>>**plotam(‘***filename.i***’,’***filename.q***’);**

This command calls the plotam.m script file. It is used to view the time averaged (as opposed to instantaneous) field amplitude constructed from two plot files differing by one-quarter cycle (the in-phase and quadrature field plots). filename.i and filename.q are instantaneous field plots taken at two different times one-quarter cycle apart.

>>**plotamsten(‘***filename.i***’,’***filename.q***’,’***top_filename***’);**

This command calls the plotamsten.m script file. It is similar to the **plotam** command above except that the topography information taken from *top_filename* is stenciled in.

>>**plotam6(‘***filename***’);**

This command calls the plotam6.m script file. It is used to assemble all three components of a field at two different times (in-phase and quadrature) (six files in total) to yield the time averaged total field intensity. It is very similar to **plotam** but includes three field components (the x,y and z) instead of just one. *filename* should be the name of the files without the `.x.i`, `.x.q` etc. extensions. The script file automatically adds them. See examples in Chapter 3 for a better understanding of how this command works.

>>**plotam6sten(‘***filename***’,’***top_filename***’);**

This command calls the plotam6sten.m script file. It is used similar to the **plotam6** command above except that the topography information taken from *top_filename* is stenciled in.

5.0 Useful Hints

5.1 Material Interface at the Boundaries

If a material interface exists very close to an absorbing boundary condition the simulation may be inaccurate. There may be evanescent fields near these materials and it is not know how the boundary conditions behave in the presence of evanescent waves. To ensure accurate results leave one wavelength between any material interfaces and any absorbing boundary conditions.

5.2 Verification of Loaded Topography

If it is suspected that the loaded topography does not coincide with which the user has in mind, the user can verify the loaded topography by plotting the real part of the refractive index or the block number using the keyword **plot** as described in Section 4.

5.3 Real and Imaginary Parts of the Refractive Index

Numerical instability may occur when the imaginary part of the refractive index of a particular material block is greater than the real part. Such a situation may arise if a metal block is present. In such instances, the material type keyword “**dispersive**” should be used instead of “**index**”.

5.4 Number of Simulation Nodes

The number of simulation nodes needed in order to give accurate simulation results can be determined by the following relation:

x_simulation_nodes > (15*largest_index**x_length*)/wavelength

and the number of simulation nodes in the y- and z-directions can be determined from *x_simulation_nodes* by the ratios of *x_length*, *y_length* and *z_length*.

5.5 Periodic Structures

Always remember that periodic boundary conditions are used on all 6 sides of the domain, including the + and - z-directions.

5.6 Isolated Structures

Isolated structures are difficult to simulate because absorbing boundary conditions are needed on all 6 sides of the domain. Even though this can be accomplished with the use PML, a problem arises with regards to the illumination - it must be “clipped” to the size of the domain. True plane wave illumination is not possible when absorbing boundary conditions exist on the sides of the domain. It is only advisable to attempt full isolation when point sources are used.

5.7 PML Thickness

The thicker the PML (in nodes) the better its performance is. Anywhere from 4 to 16 nodes of PML is suggested depending on memory restrictions. Remember, PML nodes are just like other material nodes and they take up space in the simulation domain.

5.8 2D simulations

2D simulation can be run by putting only 1 node in the y direction.

5.9 Symmetry - using it properly

Illumination will also take on the symmetry of the domain - ie. off-normal plane waves cannot exist.

5.10 Proximity to Sources and Boundary Conditions

The topography should be homogeneous (i.e. one material) in the immediate vicinity of the PML material because the PML i) can only be matched to a single material and ii) is not well characterized when evanescent waves are present. Topography should be placed at least a half wavelength (1 full wavelength recommended) away from the PML material otherwis non-physical reflections may occur.

5.11 Convergence Problems

Be aware that certain topographies can scatter light strongly into directions where periodic boundary conditions exist. This may cause convergence time to increase significantly becuase the light waves must propagate a long distance, through several periods of the periodic topography.

A.0 Appendix A: Suggested File Naming Convention

The input file should have a **.in** extenstion. (ex. `hole.in`)

All files should have a prefex the same as the input file prefix.

The output file should have a **.out** extention. (ex. `hole.out`)

All plot output files should indicate which region of the domain is plotted and what data is being plotted. If only a plane of nodes is output then the following convention should be applied:

prefix.{**xy** or **yz** or **zx**}.*position*.{**e** or **h** or **ref** or **pac** or **blk**}[.{**x** or **y** or **z**}.{**i** or **q** or *time*}]

example: `hole.yz.0.e.x.i` means the in-phase component of the x component of the electric field at steady state for an yz plane of the domain located at x = 0.

example: `hole.zx.3.blk` means the block number for a zx plane of the domain located at y = 3.

example: `hole.yz.35.e.x.2.25` means the x component of the electric field after 2.25 cycles of simulation (non-steady-state plot).

It is important to follow the naming conventions for some of the MATLAB scripts to work correctly.

B.0 Appendix B: MATLAB scripts

The MATLAB scripts can be used to view the outputs from TEMPEST. They are also useful for describing the output file formats. The four most useful scripts are presented here.

plotbin.m: lets you view a single component of a field (i.e. the Ex component or the refractive index)

plotam.m: lets you view the intensity of a particular field component (i.e. adds the squares of the in-phase and quadrature fields)

plotam6.m: lets you view the total intensity of the field (i.e. Ex_real^2+Ex_imag^2+Ey_real^2+Ey_imag^2+Ez_real^2+Ez_imag^2)

vieworders.m: let you view the output file created by the orders command.

plotbin.m

```
function a=plotbin(file1)
% comment
fp1=fopen(file1);
size1=fread(fp1,3,'int')

if size1(1)==1
  nx=size1(2);
  ny=size1(3);
  xlab='y axis (nodes)';
  ylab='z axis (nodes)';
end

if size1(2)==1
  nx=size1(1);
  ny=size1(3);
  xlab='x axis (nodes)';
  ylab='z axis (nodes)';
end

if size1(3)==1
  nx=size1(1);
  ny=size1(2);
  xlab='x axis (nodes)';
  ylab='y axis (nodes)';
end

a=fread(fp1,[nx ny],'float');
if ny==1,
plot(a);
else
pcolor(a');
%axis(‘equal');
colormap(‘hot');
shading(‘flat');
colorbar;
xlabel(xlab);
ylabel(ylab);
end
title(‘Electric Field');
fclose(fp1);
```

plotam.m

```
function [a,b,c] = plotam(file1,file2)
% comment
fp1=fopen(file1);
fp2=fopen(file2);
size1=fread(fp1,3,'long');
size2=fread(fp2,3,'long');

if size1(1)==1
  nx=size1(2);
  ny=size1(3);
  xlab='y axis (nodes)';
  ylab='z axis (nodes)';
end

if size1(2)==1
  nx=size1(1);
  ny=size1(3);
  xlab='x axis (nodes)';
  ylab='z axis (nodes)';
end

if size1(3)==1
  nx=size1(1);
  ny=size1(2);
  xlab='x axis (nodes)';
  ylab='y axis (nodes)';
end

a=fread(fp1,[nx ny],'float');
b=fread(fp2,[nx ny],'float');
c=a.*a+b.*b;
pcolor(c');
shading('flat');
caxis([0 2*average(c)]);
%axis('equal');
colormap('hot');
colorbar;
xlabel(xlab);
ylabel(ylab);
title('<|E|^2>');
fclose(fp1);
fclose(fp2);
```

plotam6.m

```
function g = plotam6(file);
% comment
fp1=fopen([file '.x.i']);
fp2=fopen([file '.x.q']);
fp3=fopen([file '.y.i']);
fp4=fopen([file '.y.q']);
fp5=fopen([file '.z.i']);
fp6=fopen([file '.z.q']);
size1=fread(fp1,3,'int')
size2=fread(fp2,3,'int')
size3=fread(fp3,3,'int')
size4=fread(fp4,3,'int')
size5=fread(fp5,3,'int')
size6=fread(fp6,3,'int')

if size1(1)==1
  nx=size1(2);
  ny=size1(3);
  xlab='y axis (nodes)';
  ylab='z axis (nodes)';
end

if size1(2)==1
  nx=size1(1);
  ny=size1(3);
  xlab='x axis (nodes)';
  ylab='z axis (nodes)';
end

if size1(3)==1
  nx=size1(1);
  ny=size1(2);
  xlab='x axis (nodes)';
  ylab='y axis (nodes)';
end

a=fread(fp1,[nx ny],'float');
b=fread(fp2,[nx ny],'float');
c=fread(fp3,[nx ny],'float');
d=fread(fp4,[nx ny],'float');
e=fread(fp5,[nx ny],'float');
f=fread(fp6,[nx ny],'float');

g=a.*a+b.*b+c.*c+d.*d+e.*e+f.*f;
g=0.5*g;
pcolor(g'); shading('flat'); caxis([0 50000]); colormap('hot');

% colorbar; */
% #xlabel(xlab); */
% #ylabel(ylab); */
% #title('<|E|^2>'); */



fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
fclose(fp5);
fclose(fp6);
```

vieworders.m

```
%[M,N,orders_te,orders_tm]=vieworders(file,Mview,mview0,nview0)
%The file must have the first entry as M, the second as N and then
%contain (2N+1)(2M+1) TE orders followed by the same number of
%TM orders. (2Mview+1)(2Mview+1) orders are displayed centered about the
%(mview0,nview0)th order.
%Tom Pistor, January 99
function [M,N,orders_te,orders_tm]=vieworders(file,Mview,mview0,nview0)

fp=fopen(file);
M=fread(fp,1,'int')
N=fread(fp,1,'int')

for y=1:2*N+1,
  for x=1:2*M+1,
    orders_te(x,y)=fread(fp,1,'double')+i*fread(fp,1,'double');
‘te'
x
y
orders_te(x,y)
  end
end
for y=1:2*N+1,
  for x=1:2*M+1,
    orders_tm(x,y)=fread(fp,1,'double')+i*fread(fp,1,'double');
‘tm'
x
y
orders_te(x,y)
  end
end

mmin=-Mview+mview0;
mmax=Mview+mview0;
nmin=-Mview+nview0;
nmax=Mview+nview0;
if N==0,
  figure; subplot(2,2,1);
  bar([mmin:mmax],abs(orders_te(M+1+mmin:M+1+mmax)));
  title([file ‘ TE Magnitude']); xlabel(‘m');
  subplot(2,2,2);
  bar([mmin:mmax],contsphase(angle(orders_te(M+1+mmin:M+1+mmax))));
  title([file ‘ TE Phase']); xlabel(‘m');
  subplot(2,2,3);
  bar([mmin:mmax],abs(orders_tm(M+1+mmin:M+1+mmax)));
  title([file ‘ TM Magnitude']); xlabel(‘m');
  subplot(2,2,4);
  bar([mmin:mmax],contsphase(angle(orders_tm(M+1+mmin:M+1+mmax))));
  title([file ‘ TM Phase']); xlabel(‘m');
else
  figure; subplot(2,2,1);
mmin
mmax
nmin
nmax
  bar3new(mmin:mmax,nmin:nmax,abs(orders_te(M+1+mmin:M+1+mmax,N+1+nmin:N+1+nmax))');
  tomax=axis;
  axis([mmin-1 mmax+1 nmin-1 nmax+1 tomax(5) tomax(6)]);
  title([file ‘ TE Magnitude']); xlabel(‘m'); ylabel(‘n');
  subplot(2,2,2);

bar3new(mmin:mmax,nmin:nmax,contsphase(angle(orders_te(M+1+mmin:M+1+mmax,N+1+nmin:N+1
+nmax))'));
  tomax=axis;
  axis([mmin-1 mmax+1 nmin-1 nmax+1 tomax(5) tomax(6)]);
  title([file ‘ TE Phase’]); xlabel(‘m’); ylabel(‘n’);

  subplot(2,2,3);
  bar3new(mmin:mmax,nmin:nmax,abs(orders_tm(M+1+mmin:M+1+mmax,N+1+nmin:N+1+nmax))’);
  tomax=axis;
  axis([mmin-1 mmax+1 nmin-1 nmax+1 tomax(5) tomax(6)]);
  title([file ‘ TM Magnitude’]); xlabel(‘m’); ylabel(‘n’);

  subplot(2,2,4);

bar3new(mmin:mmax,nmin:nmax,contsphase(angle(orders_tm(M+1+mmin:M+1+mmax,N+1+nmin:N+1
+nmax))’));
  tomax=axis;
  axis([mmin-1 mmax+1 nmin-1 nmax+1 tomax(5) tomax(6)]);
  title([file ‘ TM Phase’]); xlabel(‘m’); ylabel(‘n’);
end
fclose(fp);
```

References

1 Born & Wolf, *Principles of Optics*, p. 513, Pergamon Press, 1975.

2 T. Doi, K. Tadros, B. Kuyel, and A. R. Neureuther, “Edge-profile, Materials and Protective Coating Effects on Image Quality,” *Proc. SPIE, Integrated Circuit Metrology, Inspection and Process Control V*, vol. 1464, Mar. 1991.

3 Roberto Guerrieri, Karim H. Tadros, John Gamelin, and Andrew Neureuther, “Massively Parallel Algorithms for Scattering in Optical Lithography,” *IEEE Trans. on CAD*, vol. 10, no. 9, Sep. 1991.

4 W. G. Oldham, S. N. Nandgaonkar, A. R. Neureuther, and M. M. O’Toole, “A General Simulator for VLSI Lithography and Etching Processes: Part I - Application to Projection Lithography,” *IEEE Trans. on Electron Devices*, vol. ED-26, no. 4, pp. 717-722, Apr. 1979.

5 *SPLAT v5.0 Users’ Guide*, Memorandum no. M95/13, ERL, University of California, Berkeley, 1995.

6 K. Tadros, A. R. Neureuther, J. Gamelin, and R. Guerrieri, “Investigation of Reflective Notching with Massively Parallel Simulation,” *Proc. SPIE, Optical/Laser Microlithography III*, vol. 1264, pp. 322-332, Mar. 1990.

7 K. Tadros, A. R. Neureuther, and R. Guerrieri, “Understanding Metrology of Polysilicon Gates through Reflectance Measurements and Simulation,” *Proc. SPIE, Integrated Circuit Metrology, Inspection and Process Control V*, vol. 1464, Mar. 1991.

8 *CM5 Technical Summary*, Thinking Machines Corporation, Nov. 1992.

9 Kenny Toh, “Two-Dimensional Images with Effects of Lens Aberrations in Optical Lithography,” *M. S. Thesis*, Memorandum No. UCB/ERL M88/30, University of California, Berkeley, May 1988.

10 Kenny Toh, “Algorithms for Three-Dimensional Simulation of Photoresist Development,” Memorandum. No. UCB/ERL M90/123, *Ph.D. Dissertation*, University of California, Berkeley, December 14, 1990

11 A. Wong, T. Doi, D. Dunn, and A. R. Neureuther, “Experimental and Simulation Studies of Alignment Marks,” *Proc. SPIE, Optical/Laser Microlithography IV*, vol. 1463, Mar. 1991.

12 Alfred K. Wong, and Andrew R. Neureuther, “Polarization Effects in Mask Transmission”, *Proc. SPIE, Optical/Laser Microlithography*, Mar. 1992.

13 Alfred K. Wong, and Andrew R. Neureuther, “Edge Effects in Phase-shifting Masks for 0.25 µm Lithography,” *Proc. SPIE*, vol. 1809, pp. 222-228, 1992.

14 Alfred K. Wong, and Andrew R. Neureuther, “Mask Topography Effects in Projection Printing of Phase-Shifting Masks,” *IEEE Trans. Elec. Dev*., vol. 41, no. 6, pp. 895-902, June 1994.